# APPENDIX B. KIT-V850E2/MN4-TP INTERNAL COMMANDS

This appendix describes the KIT-V850E2/MN4-TP internal commands. These commands can be used as through commands in the debugger. For an explanation of using through commands, refer to the manual provided with the debugger.

With GHS-Multi

The through commands can be directly input in the target window after RTESERV2 has been connected.

## Command list

**Note:** These commands can be used only if the debugger does not provide equivalent functions. If these commands are issued when the debugger provides equivalent functions, a contention may occur between KIT-V850E2/MN4-TP and the debugger, causing either device to malfunction.

## Command format

The basic command format for the KIT-V850E2/MN4-TP internal commands is described below:

command-name parameter(s)

* In parameter syntax, a parameter enclosed in brackets ([ ]) is omissible. A horizontal line (|) indicates that one of the parameters delimited by it must be selected.

A command name must be an alphabetic character string, and be separated from its parameter(s) by a space or tab. A parameter must be an alphabetic character string or hexadecimal number, and be delimited by a space or tab. (A hexadecimal number cannot contain operators.)

## abp, abp1,abp2,abp3,and abp4 commands

[Format]

```
abp            [or[12]] [or34]
abp{1|2|3|4}   [ADDR [AMASK]] [data DATA [DMASK]] [asid ASID|noasid] [aeq|aneq] [deq|dneq]
               [read|write|accs] [byte|hword|word|nosize]
abp{1|2|3|4}   /del
```

[Parameters]

| | |
|---|---|
| abp [or[12]] [or34]: | The combination of abp1 and abp2 or the combination of abp3 and abp4 is set up. |
| or\|or12: | Break occurs if either abp1 or abp2 occurs. |
| or34: | Break occurs if either abp3 or abp4 occurs. |
| abp{1\|2\|3\|4}: | Input before the condition of abp1 to abp4 is specified. |
| ADDR [AMASK]: | Specifies an address condition. |
| ADDR: | Specifies addresses in hexadecimal number. |
| AMASK: | Specifies the mask data of an address in hexadecimal. Bits that are 1 will not be compared. |
| data DATA [DMASK]: | Specifies a data condition. |
| DATA: | Specifies data in hexadecimal number. |
| DMASK: | Specifies the mask data of data in hexadecimal. Bits that are 1 will not be compared. |
| asid ASID\|noasid: | For future expansion. Use “noasid". |
| aeq\|aneq: | Specifies an address comparison condition. |
| aeq: | Compares address for equality. |
| aneq: | Compares address for non-equality. |
| deqldneq: | Specifies a data comparison condition. |
| deq: | Compares data for equality. |
| dneq: | Compares data for non-equality. |
| read\|write\|accs: | Specifies a cycle condition. |
| read: | Specifies a read cycle. |
| write: | Specifies a write cycle. |
| accs: | Specifies a read or write cycle. |
| byte\|hword\|word\|nosize: | Specifies access size. |
| byte: | Specifies byte access (8 bits). |
| hword: | Specifies half-word access (16 bits). |
| word: | Specifies word access (32 bits). |
| nosize: | Specifies invalidity. |
| abp{1\|2\|3\|4} /del: | Clears a condition. |
| /del: | Specifies deletion of a condition. |

[Function]

These commands set or delete access breakpoints.
Up to four access breakpoints can be set.
They can specify execution addresses.

[Examples]

```
abp or
    abp1 or abp2 is specified.
abp2 1000 data 5555 0 aeq deq read hword
    Break occurs when 5555h is read in hword from address 1000h.
abp1 /del
    The condition set by abp1 is deleted.
```

[Remarks]

The resources of the abp command are the same as the resources of the wp command.
The resources currently used by the wp command cannot be used by the abp command.

## env and ememstat commands

[Format]

```
env   [[!]auto] [[!][verify]] [jtag[xxx][.[yyy]]{M|K}]
      [[!]resetz] [[!]hldrqz] [[!]stopz] [[!]waitz] [romless|single0] [[!]svstop]
ememstat
```

[Parameters]

[!]auto: If a breakpoint is set during execution, the breakpoint causes a temporary break. Choose [auto] to automatically perform the subsequent execution. Choose [!auto] to suppress it.

[!]verify: Specifies whether the verification after writing memory is set. Enter '!' if it is not to be set.

[jtag[xxx][.[yyy]]{M|K}]: The frequency of a JTAG clock is set up. The unit of frequency is MHz or KHz. Frequency can set up any value from 10kHz to 125MHz. But it is rounded by the following values. Actual set value can be checked by the display of the env command.

For RTE-2000-TP:

[25MHz,12.5MHz,5MHz,2MHz,1MHz,500KHz,250KHz,100KHz]

For RTE-2000H-TP:

[125MHz, 100MHz, 80MHz, 60MHz, 50MHz, 40MHz, 30MHz, 25MHz,12.5MHz, 5MHz, 2MHz, 1MHz, 500KHz, 250KHz, 100KHz, 50KHz,25KHz, 10KHz]

**Remark:** Usually, use 25MHz or 12.5MHz. If the frequency lower than 1 MHz is specified, the debugger might be slowed down in operation speed or might malfunction. A default is set automatically as the highest frequency which operates within 25MHz. When setting up the frequency more than a default, you have to set up the frequency which CPU approved. Operation of a debugger is not normal when frequency is outside the specification of CPU.

[!]resetz: Specifies whether the RESET signal is to be masked.Enter '!' if it is not to be masked.

[!hldrqz: Specifies whether the HLDRQ signal is to be masked.Enter '!' if it is not to be masked.

[!]stopz: Specifies whether the STOP signal is to be masked.Enter '!' if it is not to be masked.

[!]waitz: Specifies whether the WAIT signal is to be masked.Enter '!' if it is not to be masked.

[!]svstop: A SVSTOP function is used when CPU takes a break.Enter '!' if it do not use a function. A SVSTOP function stops count operation of a timer, serial interface, and an A/D converter.

**Remark:** In the case of CPU of a dual core, count operation will be stopped if one of cores take a break.

[Function]

The env command sets the emulation environment and displays the DCU status.
Enter only those parameters that need to be changed.   Parameters may be entered in any order.
If the same parameter is entered twice, only the last entry is valid.
The ememstat command displays the mounting status of the EMEM board.
Display examples are shown below:

```
>env
Probe:
  Unit       : RTE-2000-TP          << Displays the main unit connected.
  Rom Probe  : (use ememstat command)
  Emem Size  : (use ememstat command)
CPU Settings:
  Auto Run    = ON (auto)
  JTAGCLOCK   = 25MHz (jtag25)
  Verify      = verify off (!verify)
```

```
    CPU Mode      = single (single)
    RAM Size      = 60K (ram60)

Signals Mask:
    RESETZ        = NO MASK (!resetz)
    HLDRQZ        = NO MASK (!hldrqz)
    STOPZ         = NO MASK (!stopz)
    WAITZ         = NO MASK (!waitz)
Others:
    SVSTOP        = count continue (!svstop)


>ememstat
Board_num    EMEM_Size    ROM_Probe
  ===================================
        ROM1          32Mbyte          Extend Type 2K   << The mounting state of an EMEM board
```

[Examples]

```
env resetz
      RESET is masked.
env verify
      Sets the Verify function to ON.
env jtag40m
      Sets the JTAG clock is 40MHz.
```

## emode command

[Format]

emode

[Parameter]

None

[Function]

The emode command displays the event setting status.

[Example]

The initial status is shown below as an example:

```
Event Condition Settings:      <<Displays the setting status of the evt command.
  evt brk      !seq
  evt seqclr   !seq
  evt seq1     !seq
  evt seq2     !seq
  evt seq3     !seq
  evt seq4     !seq
  evt secon    !seq
  evt secoff   !seq
  evt qualify  !seq
  evt tout     !seq
  evt match    !seq
Event Settings (execute):      <<Displays the setting status of the eve command.
      ch Address    ASID    Cmp
  eve   1 /del
  eve   2 /del
  eve   3 /del
  eve   4 /del
  eve   5 /del
  eve   6 /del
  eve   7 /del
  eve   8 /del
Event Settings (access):       <<Displays the setting status of the eva command.
      ch Address          Data        D_Mask      ASID       A_Cmp D_Cmp Kind   Size
  eva   1 /del
  eva   2 /del
  eva   3 /del
  eva   4 /del
  eva   5 /del
  eva   6 /del
Sequence Condition Settings:   <<Displays the setting status of the seq command.
  seq 1 step4
```

## eva command

[Format]

```
eva     {1..6} [ADDR] [data DATA [MASK]] [asid ASID|noasid] [eq|lt|gt|neq|lte|gte|ign] [deq|dneq]
        [read|write|accs] [byte|hword|word|nosize] [/del]
```

[Parameters]

| | |
|---|---|
| eva {1..6}: | Specifies an access event channel (1 to 6). |
| ADDR: | Specifies the address in hexadecimal. |
| data DATA [MASK]: | Specifies a data condition. |
| DATA: | Specifies data in hexadecimal. |
| MASK: | Specifies mask data for the data in hexadecimal. Bits that are 1 will not be compared. |
| asid ASID\|noasid: | A setup of ASID conditions |
| ASID: | Specifies ASID value in hexadecimal. |
| noasid: | ASID is not used for conditions. |
| eq\|lt\|gt\|neq\|lte\|gte\|ign: | |
| eq: | Specifies that the condition is satisfied when the event address is equal to the address specified for ADDR. |
| lt: | Specifies that the condition is satisfied when the event address is smaller than the address specified for ADDR. |
| gt: | Specifies that the condition is satisfied when the event address is greater than the address specified for ADDR. |
| neq: | Specifies that the condition is satisfied when the event address is not equal to the address specified for ADDR. |
| lte: | Specifies that the condition is satisfied when the event address is smaller than or equal to the address specified for ADDR. |
| gte: | Specifies that the condition is satisfied when the event address is greater than or equal to the address specified for ADDR. |
| ign: | Specifies that ADDR is not used as a comparison condition. |
| deq\|dneq: | Specifies a data comparison condition. |
| deq: | Compares data for equality. |
| dneq: | Compares data for non-equality. |
| read\|write\|accs: | Specifies a cycle condition. |
| read: | Specifies a read cycle. |
| write: | Specifies a write cycle. |
| accs: | Specifies a read or write cycle. |
| byte\|hword\|word\|nosize: | Specifies access size. |
| byte: | Specifies byte access (8 bits). |
| hword: | Specifies half-word access (16 bits). |
| word: | Specifies word access (32 bits). |
| nosize: | Specifies invalidity. |
| eva {1..6} /del: | Clears a condition. |
| /del: | Specifies deletion of a condition. |

[Function]

The eva command sets an access event. The specified event can be combined with a condition using the evt command to be used as a break or trace condition.

[Examples]

eva 1 ffff000 data 55 00 byte read

A cycle for reading 0x55 starting at address 0xffff000 is set for eva ch1 with using the default values for other parameters.

eva 1 /del

The condition of eva ch1 is cleared.

[Remarks]

When using an event on trace conditions, the data conditions of the read cycle conditions are ignored. The trace conditions in that case are the read cycles which are in agreement except for data conditions.

## eve command

[Format]

```
eve   {1..8} [ADDR] [asid ASID|noasid] [eq|lt|gt|neq|lte|gte|ign] [/del]
```

[Parameters]

| | | |
|---|---|---|
| eve | {1..8}: | Specifies an execution event channel (1 to 8). |
| | ADDR: | Specifies the address in hexadecimal. |
| asid ASID\|noasid: | | A setup of ASID conditions |
| | ASID: | Specifies ASID value in hexadecimal. |
| | noasid: | ASID is not used for conditions. |
| eq\|lt\|gt\|neq\|lte\|gte\|ign: | | |
| | eq: | Specifies that the condition is satisfied when the event address is equal to the address specified for ADDR. |
| | lt: | Specifies that the condition is satisfied when the event address is smaller than the address specified for ADDR. |
| | gt: | Specifies that the condition is satisfied when the event address is greater than the address specified for ADDR. |
| | neq: | Specifies that the condition is satisfied when the event address is not equal to the address specified for ADDR. |
| | lte: | Specifies that the condition is satisfied when the event address is smaller than or equal to the address specified for ADDR. |
| | gte: | Specifies that the condition is satisfied when the event address is greater than or equal to the address specified for ADDR. |
| | ign: | Specifies that ADDR is not used as a comparison condition. |
| eve {1..8} /del: | | Clears a condition. |
| | /del: | Specifies deletion of a condition. |

[Function]

The eve command sets an execution event. The specified event can be combined with a condition using the evt command to be used as a break or trace condition.

[Examples]

eve 1 1000

Execution of the instruction at address 0x1000 is set for eve ch1 using the default values for other parameters.

eve 1 /del

The condition of eve ch1 is cleared.

## evt command

[Format]

```
evt   {brk|seqclr|seq1|seq2|seq3|seq4|secon|secoff|qualify|tout|match}
      [evep{[1][2][3]..[8]}] [ever{[1][3][5][7]}] [evap{[1][2][3]..[6]}] [evar{[1][3][5]}]
      [wp{[1][2][3][4]}] [[!]seq] [[!]evtin]
```

[Parameters]

brk|seqclr|seq1|seq2|seq3|seq4|secon|secoff|qualify|tout|match:

Specifies a condition with which the event is to be combined.

| | |
|---|---|
| brk: | Specifies a break condition. |
| seqclr: | Specifies a sequential clear condition. |
| seq1: | Specifies a first-step sequential condition. |
| seq2: | Specifies a second-step sequential condition. |
| seq3: | Specifies a third-step sequential condition. |
| seq4: | Specifies a fourth-step sequential condition. |
| secon: | Specifies a trace section on condition. |
| secoff: | Specifies a trace section off condition. |
| qualify: | Specifies a trace qualify condition. |
| tout: | Specifies a trigger output condition. |
| match: | Specifies a trace trigger condition. |
| evep{[1][2][3]..[8]}: | Specifies the corresponding event specified by the eve command as a point by itself. Specifying this parameter with no numeric characters cancels the setting. |
| [1][2][3]..[8]: | Each number corresponds to a channel number specified by eve. |
| ever{[1][3][5][7]}: | Specifies each pair of events specified by the eve command as an area. Specifying this parameter with no numeric characters cancels the setting. |
| 1: | Specifies the conditions of channels 1 and 2 specified by eve as a range (and condition). |
| 3: | Specifies the conditions of channels 3 and 4 specified by eve as a range (and condition). |
| 5: | Specifies the conditions of channels 5 and 6 specified by eve as a range (and condition). |
| 7: | Specifies the conditions of channels 7 and 8 specified by eve as a range (and condition). |
| evap{[1][2][3]..[6]}: | Specifies the corresponding event specified by the eva command as a point by itself.<br>Specifying this parameter with no numeric characters cancels the setting. |
| [1][2][3]..[6]: | Each number corresponds to a channel number specified by eva. |
| evar{[1][3][5]}: | Specifies each pair of events specified by the eva command as an area. Specifying this parameter with no numeric characters cancels the setting. |
| 1: | Specifies the conditions of channels 1 and 2 specified by eva as a range (and condition). |
| 3: | Specifies the conditions of channels 3 and 4 specified by eva as a range (and condition). |
| 5: | Specifies the conditions of channels 5 and 6 specified by eva as a range (and condition). |
| wp{[1][2][3][4]}: | Specifies the corresponding event specified by the wp command as a point by itself. Specifying this parameter with no numeric characters cancels the setting. |
| [1][2][3][4]: | Each number corresponds to a point number specified by wp. |

| | |
|---|---|
| [!]seq: | Specifies a sequential condition. |
| seq: | Specifies a sequential condition. Enter ‘!’ to cancel the sequential condition. ‘!’ cannot be specified for a seq-related condition (seqclr, seq1, seq2, seq3, or seq4). |
| [!]evtin: | Specifies the external event detection condition. Enter ‘!’ to cancel the external event detection condition. |

[Function]

The evt command specifies the use of each event specified by eve command, eva command , and wp command.

[Remark]

- The evap and the evar parameter cannot be specified with the seqon, the secoff, and the qualify.
- The wp parameter can be specified only with the tout.
- When you use seq, please specify only the event of evep or ever as seqclr,seq1,seq2,se3, and seq4.
- The evtin parameter can be used with brk, secon, secoff, tout, and match.
- For the details of the trace section and qualify conditions, see Appendix A, "Details of Trace Functions".

[Examples]

evt brk evep1234 ever5 evap12 evar3

As break events, the events specified for channels 1 to 4 by eve are used as points; those specified for channels 5 and 6 as a range condition; those specified for channels 1 and 2 by eva as points; and those specified for channels 3 and 4 as a range.

evt brk evep ever evap evar

The events specified for evep, ever, evap, and evar as break events are canceled.

## evtinenv command

[Format]

```
evtinenv    {inpe1|inpe2|indma|evto} [[!]toutpe1] [[!]toutpe2] [[!]toutdma]] [[!]evti]
            [toutclk{1|2|3|4}] [dmatoutclk{1|2|3|4}]
```

[Parameters]

| | |
|---|---|
| inpe1\|inpe2\|indma\|evto: | Specifies the generation cause to the event integration part. |
| inpe1: | Specifies the generation cause of the event input to CPU(PE1). |
| inpe2: | Specifies the generation cause of the event input to CPU(PE2). |
| indma: | Specifies the generation cause of the event input to DMA |
| evto: | Specifies the generation cause of the event trigger output to the external debugging device. |
| [!]toutpe1: | Specifies the event trigger output request by PE1 for a generation cause. It is not set when '!' is entered. |
| [!]toutpe2: | Specifies the event trigger output request by PE2 for a generation cause. It is not set when '!' is entered. |
| [!]toutdma: | Specifies the event trigger output request by DMA for a generation cause. It is not set when '!' is entered. |
| [!]evti: | Specifies the external event input from the external debugging device for a generation cause. It is not set when '!' is entered. |
| toutclk{1\|2\|3\|4}: | Specifies the ratio of dividing frequency to the event trigger output of CPU. Each parameter is equal to 1/1, 1/2, 1/3, and 1/4. |
| dmatoutclk{1\|2\|3\|4}: | Specifies the ratio of dividing frequency of the event trigger output of DMA. Each parameter is equal to 1/1, 1/2, 1/3, and 1/4. |

[Function]

The evtinenv command specifies the generation cause of the event input to the event integration part specified by evt and dmaevt.

[Remark]

The parameter cannot be specified by the following combinations:

- inpe1 and toutpe1
- inpe2 and toutpe2
- indma and toutdma
- evto and evti

## dmaeva command

[Format]

```
dmaeva    {1..6} [ADDR] [data DATA [MASK]] [eq|lt|gt|neq|lte|gte|ign] [deq|dneq]
          [exec|read|write|accs] [byte|hword|word|nosize] [ch CHNO|allch]
          [[!]evtin] [[!]csext] [[!]cspe1] [[!]cspe2] [/del]
```

[Parameters]

| | |
|---|---|
| dmaeva {1..6}: | Specifies an DMA event channel (1-6). |
| ADDR: | Specifies the address in hexadecimal. |
| data DATA [MASK]: | Specifies a data condition. |
| DATA: | Specifies data in hexadecimal. |
| MASK: | Specifies mask data for the data in hexadecimal. Bits that are 1 will not be compared. |
| eq\|lt\|gt\|neq\|lte\|gte\|ign: | |
| eq: | Specifies that the condition is satisfied when the event address is equal to the address specified for ADDR. |
| lt: | Specifies that the condition is satisfied when the event address is smaller than the address specified for ADDR. |
| gt: | Specifies that the condition is satisfied when the event address is greater than the address specified for ADDR. |
| neq: | Specifies that the condition is satisfied when the event address is not equal to the address specified for ADDR. |
| lte: | Specifies that the condition is satisfied when the event address is smaller than or equal to the address specified for ADDR. |
| gte: | Specifies that the condition is satisfied when the event address is greater than or equal to the address specified for ADDR. |
| ign: | Specifies that ADDR is not used as a comparison condition. |
| deq\|dneq: | Specifies a data comparison condition. |
| deq: | Compares data for equality. |
| dneq: | Compares data for non-equality. |
| read\|write\|accs: | Specifies a cycle condition. |
| read: | Specifies a read cycle. |
| write: | Specifies a write cycle. |
| accs: | Specifies a read or write cycle. |
| byte\|hword\|word\|nosize: | Specifies access size. |
| byte: | Specifies byte access (8 bits). |
| hword: | Specifies half-word access (16 bits). |
| word: | Specifies word access (32 bits). |
| nosize: | Specifies invalidity. |
| ch CHNO\|allch: | Specifies an DMA channel condition |
| CHNO: | Specifies the DMA channel number in hexadecimal.<br>0...7F: DTS channnel 0...127<br>80...FE: DMAC channel 0...126 |
| allch: | Specifies that doesn't use the DMA channel condition. |
| [!]evtin: | Specifies the external event input condition.<br>It is not set when '!' is entered. |
| [!]csext: | Specifies the chip selection condition (external resource).<br>It is not set when '!' is entered. |
| [!]cspe1: | Specifies the chip selection condition (CPU(PE1) access).<br>It is not set when '!' is entered. |
| [!]cspe2: | Specifies the chip selection condition (CPU(PE2) access).<br>It is not set when '!' is entered. |
| eva {1..6} /del: | Clears a condition. |
| /del: | Specifies deletion of a condition. |

[Function]

Set the event of the DMA access functions. The specified event can be integrated by the dmaevt command, and be used as a condition of the break and the trace.

[Examples]

```
dmaeva 1 100000 data 55 00 byte read
            A cycle for reading 0x55 starting at address 0x100000 is set for dmaeva ch1.
dmaeva 1 /del
             The condition of dmaeva ch1 is cleared.
```

## dmaevt command

[Format]

```
dmaevt    {brk|tout|match|secon|secoff|qualify}
          [evap{[1][2][3]..[6]}] [evar{[1][3][5]}] [[!]evtin]
```

[Parameters]

brk|seqclr|seq1|seq2|seq3|seq4|secon|secoff|qualify|tout|match:

Specifies a condition with which the event is to be combined.

| | |
|---|---|
| brk: | Specifies a break condition. |
| secon: | Specifies a trace section on condition. |
| secoff: | Specifies a trace section off condition. |
| qualify: | Specifies a trace qualify condition. |
| tout: | Specifies a trigger output condition. |
| match: | Specifies a trace trigger condition. |
| evap{[1][2][3]..[6]}: | Specifies the corresponding event specified by the dmaeva command as a point by itself. Specifying this parameter with no numeric characters cancels the setting. |
| [1][2][3]..[6]: | Each number corresponds to a channel number specified by dmaeva. |
| evar{[1][3][5]}: | Specifies each pair of events specified by the eva command as an area. Specifying this parameter with no numeric characters cancels the setting. |
| 1: | Specifies the conditions of channels 1 and 2 specified by eva as a range (and condition). |
| 3: | Specifies the conditions of channels 3 and 4 specified by eva as a range (and condition). |
| 5: | Specifies the conditions of channels 5 and 6 specified by eva as a range (and condition). |
| [!]evtin: | Specifies the external event detection condition. It is not set when '!' is entered. |

[Function]

The dmaevt command specifies the use of each event specified by dmaeva.

[Examples]

dmaevt brk evap13 ever5

As break events, those specified for channels 1 and 3 by dmaeva as points; and those specified for channels 5 and 6 as a range.

dmaevt brk evap evar

The events specified for evep, and evar as break events are canceled.

## help command

[Format]

```
help   [command]
```

[Parameter]

| | |
|---|---|
| command: | Specifies the name of the command for which you require help.<br>If this parameter is omitted, a list of commands is displayed. |

[Function]

The help command displays a help message for a specified command.

[Example]

```
help env
```

A help message for the env command is displayed.

## ifrom command

[Format]

```
Ifromenv     [[!]write] [[!]verify] [[!]tracecache]
ifromclear
ifromflush
ifromrefill
```

[Parameters]

[[!]write]: Specifies automatic writing mode to internal flash ROM

| | |
|---|---|
| write: | mode is ON |
| !write: | mode is OFF |

Initial value is write.

[[!]verify]: Specifies verify operation after writing when automatic writing mode is ON.

| | |
|---|---|
| verify: | do verify operation. |
| !verify: | no verify operation will be done. |

Initial value is !verify.

[[!]tracecache]: The data of an internal ROM area are saved. Save data are used when displaying trace during program execution.

| | |
|---|---|
| tracecache: | save. |
| !tracecache: | not save. |

Initial value is tracecache.

[Function & examples]

ifromenv write !verify

Setting automatic writing mode.

Automatic writing mode is set and verify is off in this case. All the data will be written when program will be execution. The writing time of an internal flash ROM may be long.

ifromclear

Set cache state to INVALID when automatic writing mode.

ifromflush

Write cache data of the DIRTY block to internal flash ROM when automatic writing mode.

Ifromrefill

All the data of an IROM area are cached. In this case, write or tracecache parameter of the ifromenv command must be available.

[Note]

Refer to the chapter 6 “Internal flash ROM support”.

## inb, inh, and inw commands

[Format]

inb [ADDR]
inh [ADDR]
inw [ADDR]

[Parameter]

ADDR: Specifies the address of an input port in hexadecimal.

[Function]

The inb, inh, and inw commands read the memory in different sizes.
inb, inh, and inw are the access sizes of a byte, a halfword, and word, respectively.

[Examples]

inb 1000
Memory is read in bytes (8-bit units), starting at 1000H.
inh 1000
Memory is read in half words (16-bit units), starting at 1000H.
inw 1000
Memory is read in words (32-bit units), starting at 1000H.

## init command

[Format]

init

[Parameter]

None

[Function]

The init command initializes KIT-V850E2/MN4-TP. When CPU is a dual core, both of cores are initialized. All environment values are initialized. A memory cache rejection area is not initialized.

## jread command

[Format]

```
jread  [ADDR [LENGTH]]
```

[Parameters]

| | |
|---|---|
| ADDR: | Specifies an address in hexadecimal. |
| LENGTH: | Specifies the number of bytes to be read, in hexadecimal. (Max.:  100h) |

[Function]

The jread command reads the ROM emulation area allocated by the ROM command, via JTAG (the CPU).
(Access to the ROM emulation area by ordinary commands is performed directly on internal memory.)

[Example]

```
jread 100000 100
```

100h bytes, starting at 100000h, are read via JTAG.

## nc command

[Format]

nc [[ADDR [LENGTH]]

[Parameters]

| | |
|---|---|
| ADDR: | Specifies the start address of a memory cache rejection area. |
| LENGTH: | Specifies the length of the memory cache rejection area in bytes. The default value is 32 bytes. The allowable minimum value is also 32 bytes. |

[Function]

To ensure quick memory access, KIT-V850E2/MN4-TP provides a memory read cache. When the same memory address is accessed more than once, the read operation is not actually performed. This cache operation conflicts with the actual operation when an I/O unit is mapped onto memory. In such a case, specify a memory cache rejection area by using the nc command. Up to eight blocks can be specified as a memory cache rejection area. The allowable minimum block size is 32 bytes.

[Example]

The initial status is shown below as an example:

```
>nc
 No Memory Cache Area
 No  Address    Length
  1  f0000000   0e9fffff
  2  fee00000   011fffff
```

## ncd command

[Format]

ncd block-number

[Parameter]

block-number: Specifies the block number for a memory cache rejection area to be deleted.

[Function]

The ncd command deletes a memory cache rejection area. Specify the block number corresponding to the memory cache rejection area to be deleted. Do not delete any default memory cache rejection area.

If an default memory cache rejection area is deleted, accessing an I/O space by a command may fail to read correct values.

[Examples]

ncd 1

Block 1 is deleted from the memory cache rejection area.

>>This is just an example. Do not delete the block actually.

```
>nc
 No Memory Cache Area
 No  Address   Length
   1  f0000000  0e9fffff
   2  fee00000  011fffff

>ncd 2
 No Memory Cache Area
 No  Address   Length
   1  f0000000  0e9fffff
```

## nsbp command

[Format]

```
nsbp  [[ADDR [LENGTH]]
```

[Parameters]

| | |
|---|---|
| ADDR: | Specifies the start address of a software break prohibition area. |
| LENGTH: | Specifies the length of a software break prohibition area in bytes.<br>The minimum unit of a specification area is the boundary of half word.<br>The number of the areas which can be specified is a maximum of four. |

[Function]

The nsbp command specifies an area to forbid a software break.
When a breakpoint is specified, a debugger implicitly performs a memory test (write access) to an object address.
The state of some flash ROM may change by performing write access and right data may not be read.
When this happens, please forbid a software break by this command to prohibit use of write cycles.
Usually, it is not necessary to specify.

[Example]

nsbp 10000 20000

A 20000-byte area, starting at 10000h, is specified as a software break prohibition area.

```
>nsbp 100000 20000
Num  Address    Length
01   00100000   00020000
```

## nsbpd command

[Format]

nsbpd [block-number|/all]

[Parameters]

| | |
|---|---|
| block-number: | Specifies the block number of the software break prohibition area to be deleted. |
| /all: | Specifies all software break prohibition area to be deleted. |

[Function]

The nsbpd command deletes the software break prohibition area specified by nsbp.

[Examples]

nsbpd 1

Block 1 is deleted from a software break prohibition area.

```
>nsbp
Num  Address   Length
01   00100000  00200000
02   00400000  00010000

>nsbpd 1
Num  Address   Length
01   00400000  00010000
```

## nrom command

[Format]

```
nrom    [[ADDR [LENGTH]]
```

[Parameters]

| | |
|---|---|
| ADDR: | Specifies the start address of a forced user area. |
| LENGTH: | Specifies the length of a forced user area in bytes. |
| | The minimum unit of the a specification area is as follows. |
| | Depends on the size of the ROM being emulated. |
| | 8/16 bits: 128K bytes |
| | 32 bits: 256K bytes |
| | (64 bits: 512K bytes) |
| | The number of areas which can be specified is a maximum of four. |

[Function]

The nrom command specifies the area when part of ROM emulation area specified by ROM command is mapped to other resources on a user system. Usually, it is not necessary to specify this command.

The operations for the specified area are as follows.

- An access from the debugger is forcibly made to the user system.
- The EMEMEN- signal of the ROM cable is deasserted inactive (high level) in the cycle for accessing this area during execution.

[Examples]

nrom 0 20000

A 20000-byte area, starting at 0h, is specified as a forced user area.

```
>nrom 0 20000
No.    Address    Length
  1    00000000   00020000

>nrom 100000 40000
No.    Address    Length
  1    00000000   00020000
  2    00100000   00040000
```

## nromd command

[Format]

nromd [block-number|/all]

[Parameters]

| | |
|---|---|
| block-number: | Specifies the block number for the forced user area to be deleted. |
| /all: | Specifies all the forced user area to be deleted. |

[Function]

The nromd command deletes the forced user area specified by nrom.

[Examples]

nromd 1

Block 1 is deleted from the forced user area.

```
>nrom 100000 40000
No.     Address     Length
  1     00000000    00020000
  2     00100000    00040000

>nromd 1
No.     Address     Length
  1     00100000    00040000
```

## outb, outh, and outw commands

[Format]

```
outb     [[ADDR] DATA]
outh     [[ADDR] DATA]
outw     [[ADDR] DATA]
```

[Parameters]

| | |
|---|---|
| ADDR: | Specifies the address of an output port in hexadecimal. |
| DATA: | Specifies the data to be output in hexadecimal. |

[Function]

The outb, outh, and outw commands write data to the memory in different sizes.
outb, outh, and outw are the access sizes of a byte, a halfword, and word, respectively.

[Examples]

outb 1000 12
Byte data 12h is written to 1000H in the memory.

outh 1000 1234
Half word data 1234h is written to 1000H in the memory.

outw 1000 12345678
Word data 12345678h is written to 1000H in the memory.

## reset command

[Format]
reset

[Parameter]
None

[Function]
The reset command resets the CPU.

## rom1..rom4 commands

[Format]

```
rom1  [ADDR [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16|bus32|bus64] [[!]wren]
rom2  [ADDR [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16] [[!]wren]
rom3  [ADDR [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16|bus32] [[!]wren]
rom4  [ADDR [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16] [[!]wren]
```

rom1: This command performs setting of a module including the EMEM board mounted to slot #3.
rom2: This command performs setting of a module including the EMEM board mounted to slot #4.
rom3: This command performs setting of a module including the EMEM board mounted to slot #5.
rom4: This command performs setting of a module including the EMEM board mounted to slot #6.

[Parameters]

ADDR [LENGTH]: Specifies an area to be emulated.

ADDR: Specifies a start address. An error occurs if the specified start address does not match the lowest address of the ROM to be emulated (boundary of the ROM).

LENGTH: Specifies the number of bytes of the ROM to be emulated.

Remark: The minimum unit of the specification area is as follows. Depends on the size of the ROM being emulated.

| | |
|---|---|
| - 8/16 bits: | 128K bytes |
| - 32 bits: | 256K bytes |
| - 64 bits: | 512K bytes |

512k|1m|2m|4m|8m|16m|32m|64m|128m|256m:
Specifies the bit size of the ROM to be emulated.
Sizes from 512K to 256M bits can be specified. For the 27C1024, for example, specify 1M bit.

rom8|rom16: Specifies the number of data bits of the ROM to be emulated.
Either 8 bits or 16 bits can be specified. If a DIP-32-ROM cable is used, choose rom8; if a DIP-40/42-ROM or STD-16BIT-ROM cable is used, choose rom16.

bus8|bus16|bus32|bus64:
Specifies the ROM bus size in the system to be emulated. 8 bits, 16 bits, 32 bits, or 64 bits can be specified.
>> [bus32, bus64] are parameters for future use. (They are not used with KIT-V850E2/MN4-TP.)

[[!]wren]: Write Enable: This setting is for using the emulation memory as RAM. wren enables writing, and !wren disables writing. The default value is !wren.

[Function]

The rom1 to rom4 commands set the ROM emulation environment of RTE-2000(H)-TP. ADDR and LENGTH must be input in pairs. Input other parameters only when their values need to be changed. Parameters may be entered in any order. If the same parameter is entered twice, only the last entry is valid. The initial value of LENGTH is 0 (not used).

[Examples]

rom1 100000 300000 32m rom16 bus16 !wren

| Slot position of EMEM board | Address range | Bus width | ROM | | Write enable |
|---|---|---|---|---|---|
| | | | Bus width | Bits | |
| #3 | 100000 – 3fffff | 16 bits | 16 bits | 32M bits | Disabled |

rom2 140000 40000 2m rom16 bus16 wren

| Slot position of EMEM board | Address range | Bus width | ROM | | Write enable |
|---|---|---|---|---|---|
| | | | Bus width | Bits | |
| #4 | 140000 - 17ffff | 16 bits | 16 bits | 2M bits | Enabled |

rom1 0 80000 2m rom16 bus32 !wren

| Slot position of EMEM board | Address range | Bus width | ROM | | Write enable |
|---|---|---|---|---|---|
| | | | Bus width | Bits | |
| #3 + #4 | 000000 - 07ffff | 32 bits | 16 bits | 2M bits | Disabled |

Do not issue the rom2 command at this time.

[Remark]

Note on area specified by rom command

Access to the range specified by the rom1..rom4 commands from the debugger is a direct access to the emulation memory in the tool. As a result, display is performed correctly even if the processor cannot correctly access ROM. It is therefore recommended to read and check data by using the jread command (that reads data via the CPU bus) or write data by setting verify to ON with the env command (download) in the initial stage of debugging.

Relationship between rom command and EMEM board

| rom command | Bus width | Slot position of EMEM board | Unusable rom command |
|---|---|---|---|
| rom1 | 8-bit | #3 | |
| | 16-bit | #3 | |
| | 32-bit | #3+#4 | rom2 |
| | 64-bit | #3+#4+#5+#6 | rom2, rom3, rom4 |
| rom2 | 8-bit | #4 | |
| | 16-bit | #4 | |
| rom3 | 8-bit | #5 | |
| | 16-bit | #5 | |
| | 32-bit | #5+#6 | rom4 |
| rom4 | 8-bit | #6 | |
| | 16-bit | #6 | |

## seq command

[Format]

```
seq    [PASS] [step{1|2|3|4}]
```

[Parameters]

| | |
|---|---|
| PASS: | Specifies in decimal the number of times the sequence condition is to be satisfied. |
| step{1\|2\|3\|4}: | Specifies the number of steps in the sequence. |
| step1: | seq4->pass_count_decrement |
| step2: | seq3->seq4->pass_count_decrement |
| step3: | seq2->seq3->seq4->pass_count_decrement |
| step4: | seq1->seq2->seq3->seq4->pass_count_decrement |

[Function]

The seq command sets the sequential conditions.
Use eve, eva, and evt to specify conditions for seq1 to seq4.
When the seqclr condition is satisfied during a sequence, the sequence is executed from the beginning.

[Example]

```
seq 100 step1
```

A seq event occurs when conditions seq1 -> seq2 -> seq3 -> seq4 are satisfied 100 times.

## sswon and sswoff commands

[Format]

```
ssw{on|off}   [evap{1|2|3|4|5|6} {none|read|write|accs}]
              [evar{1|3|5} {none|read|write|accs}]
              [td{1|2|3|4} {none|read|write|accs}]
              [[!]wp{1|2|3|4}] [[!]owner]
              [[!]startpoint] [[!]section] [[!]qualify] [[!]match]
              [[!]program] [[!]directbranch] [[!]cmov]
              [mainswon_wp_{[1][2][3][4]}] [mainswoff_wp_{[1][2][3][4]}]
```

[Parameters]

sswon: This command specifies a cycle in which trace data is to be loaded when the sub-switch is on.

sswoff: This command specifies a cycle in which trace data is to be loaded when the sub-switch is off.

evap{1|2|3|4|5|6} {none|read|write|accs}:
Specifies the type of cycle in which trace data is to be loaded for each point condition specified by the eva command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

evar{1|3|5} {none|read|write|accs}:
Specifies the type of cycle in which trace data is to be loaded for each range condition specified by the eva command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

td{1|2|3|4} {none|read|write|accs}:
Specifies the type of cycle in which trace data is to be loaded for each condition specified by the td command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

| | |
|---|---|
| [!]wp{1\|2\|3\|4}: | Loads trace message output on the condition of specifying it by the wp command. It is not load when ‘!’ is entered. |
| [!]owner: | Loads ownership trace message. It is not load when ‘!’ is entered. |
| [!]startpoint: | Loads detection trace message of the tsp command. It is not load when ‘!’ is entered. |
| [!]section: | Loads detection trace message of section ON/OFF. It is not load when ‘!’ is entered. |
| [!]qualify: | Loads detection trace message of the qualify event. It is not load when ‘!’ is entered. |
| [!]match: | Loads detection trace message of the match point event of the evt command. It is not load when ‘!’ is entered. |
| [!]program: | Loads trace message by discontinuous detection of the program by the branch etc. It is not load when ‘!’ is entered. |
| [!]directbranch: | Loads trace message by the direct branch. It is not load when ‘!’ is entered. |
| [!]cmov: | Loads trace message of the cmov instruction. It is not load when ‘!’ is entered. |

mainswon_wp_{[1][2][3][4]}:
Specifies the point condition of the wp command that switches the main switch to turning on.

mainswoff_wp_{[1][2][3][4]}:
Specifies the point condition of the wp command that switches the main switch to turning off.

[Function]

The sswon and sswoff commands specify the types of cycles in which trace data is to be loaded according to the sub-switch status.

[Examples]

By default, trace data in all cycles is to be loaded when the sub-switch is on and trace data in no cycles is to be loaded when it is off.
These commands can be used to control the loading of trace data under any desired conditions.
The default settings are shown below.

```
>sswon
Sub-switch ON Settings:
   Main Switch On Watch Point   = No Watch point (mainswon_wp_)
   Main Switch Off Watch Point  = No Watch point (mainswoff_wp_)
   Data Trace Point 1 Trace     = Read and Write cycle (td1 accs)
   Data Trace Point 2 Trace     = Read and Write cycle (td2 accs)
   Data Trace Point 3 Trace     = Read and Write cycle (td3 accs)
   Data Trace Point 4 Trace     = Read and Write cycle (td4 accs)
   Event Access Point 1 Trace   = Not Trace (evap1 none)
   Event Access Point 2 Trace   = Not Trace (evap2 none)
   Event Access Point 3 Trace   = Not Trace (evap3 none)
   Event Access Point 4 Trace   = Not Trace (evap4 none)
   Event Access Point 5 Trace   = Not Trace (evap5 none)
   Event Access Point 6 Trace   = Not Trace (evap6 none)
   Event Access Range 1 Trace   = Not Trace (evar1 none)
   Event Access Range 3 Trace   = Not Trace (evar3 none)
   Event Access Range 5 Trace   = Not Trace (evar5 none)
   Watch Point 1 Trace          = Disable (!wp1)
   Watch Point 2 Trace          = Disable (!wp2)
   Watch Point 3 Trace          = Disable (!wp3)
   Watch Point 4 Trace          = Disable (!wp4)
   Ownership Trace              = Disable (!owner)
   Start Point Trace            = Enable   (startpoint)
   Section Event Trace          = Enable   (section)
   Qualify Event Trace          = Enable   (qualify)
   Match Event Trace            = Enable   (match)
   Program Trace                = Enable   (program)
   Direct Branch Trace          = Enable   (directbranch)
   CMOV Trace                   = Enable   (cmov)Sub-switch ON Settings:

>sswoff
Sub-switch OFF Settings:
   Main Switch On Watch Point   = No Watch point (mainswon_wp_)
   Main Switch Off Watch Point  = No Watch point (mainswoff_wp_)
   Data Trace Point 1 Trace     = Not Trace (td1 none)
   Data Trace Point 2 Trace     = Not Trace (td2 none)
   Data Trace Point 3 Trace     = Not Trace (td3 none)
   Data Trace Point 4 Trace      = Not Trace (td4 none)
   Event Access Point 1 Trace    = Not Trace (evap1 none)
   Event Access Point 2 Trace    = Not Trace (evap2 none)
   Event Access Point 3 Trace    = Not Trace (evap3 none)
   Event Access Point 4 Trace    = Not Trace (evap4 none)
   Event Access Point 5 Trace    = Not Trace (evap5 none)
   Event Access Point 6 Trace    = Not Trace (evap6 none)
   Event Access Range 1 Trace    = Not Trace (evar1 none)
   Event Access Range 3 Trace    = Not Trace (evar3 none)
   Event Access Range 5 Trace    = Not Trace (evar5 none)
   Watch Point 1 Trace           = Disable (!wp1)
   Watch Point 2 Trace           = Disable (!wp2)
   Watch Point 3 Trace           = Disable (!wp3)
   Watch Point 4 Trace           = Disable (!wp4)
   Ownership Trace               = Disable (!owner)
   Start Point Trace             = Disable (!startpoint)
   Section Event Trace           = Disable (!section)
   Qualify Event Trace           = Disable (!qualify)
   Match Event Trace             = Disable (!match)
   Program Trace                 = Disable (!program)
   Direct Branch Trace           = Disable (!directbranch)
   CMOV Trace                    = Disable (!cmov)
```

## dmasswon, dmasswoff commands

[Format]

```
dmassw{on|off}    [evap{1|2|3|4|5|6} {none|read|write|accs}]
                  [evar{1|3|5} {none|read|write|accs}]
                  [td{1|2|3|4} {none|read|write|accs}]
                  [startpoint [none|read|write|accs]] [[!]section] [[!]qualify] [[!]match]
                  [[!]status]
```

[Parameters]

dmasswon: This command specifies DMA cycles in which trace data is to be loaded when the sub-switch is on.

dmasswoff: This command specifies DMA cycles in which trace data is to be loaded when the sub-switch is off.

evap{1|2|3|4|5|6} {none|read|write|accs}:

Specifies the type of cycle in which trace data is to be loaded for each point condition specified by the dmaeva command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

evar{1|3|5} {none|read|write|accs }:

Specifies the type of cycle in which trace data is to be loaded for each range condition specified by the dmaeva command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

td{1|2|3|4} {none|read|write|accs}:

Specifies the type of cycle in which trace data is to be loaded for each condition specified by the dmatd command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

startpoint [none|read|write|accs]]:

Specifies the type of cycle in which trace data is to be loaded for each condition specified by the dmatsp command.

| | |
|---|---|
| none: | Does not load trace data. |
| read: | Loads trace data in read cycles only. |
| write: | Loads trace data in write cycles only. |
| accs: | Loads trace data in read and write cycles. |

| | |
|---|---|
| [!]section: | Loads detection trace message of section ON/OFF.<br>It is not load when '!' is entered |
| [!]qualify: | Loads detection trace message of the qualify event.<br>It is not load when '!' is entered |
| [!]match: | Loads detection trace message of the match point event of the evt command.<br>It is not load when '!' is entered |
| [!]status: | Loads trace trace message of DMA status information.<br>It is not load when '!' is entered |

[Function]

The sswon and sswoff commands specify the types of DMA cycles in which trace data is to be loaded according to the sub-switch status.

[Examples]

By default, trace data in all cycles is to be loaded when the sub-switch is on and trace data in no cycles is to be loaded when it is off.

These commands can be used to control the loading of trace data under any desired conditions.

The default settings are shown below.

```
>dmasswon
DMA Sub-switch ON Settings:
  Data Trace Point 1 Trace       = Read and Write cycle (td1 accs)
  Data Trace Point 2 Trace       = Read and Write cycle (td2 accs)
  Data Trace Point 3 Trace       = Read and Write cycle (td3 accs)
  Data Trace Point 4 Trace       = Read and Write cycle (td4 accs)
  Event Access Point 1 Trace     = Not Trace (evap1 none)
  Event Access Point 2 Trace     = Not Trace (evap2 none)
  Event Access Point 3 Trace     = Not Trace (evap3 none)
  Event Access Point 4 Trace     = Not Trace (evap4 none)
  Event Access Point 5 Trace     = Not Trace (evap5 none)
  Event Access Point 6 Trace     = Not Trace (evap6 none)
  Event Access Range 1 Trace     = Not Trace (evar1 none)
  Event Access Range 3 Trace     = Not Trace (evar3 none)
  Event Access Range 5 Trace     = Not Trace (evar5 none)
  Start Point Trace              = Read and Write cycle (startpoint accs)
  Section Event Trace            = Enable   (section)
  Qualify Event Trace            = Enable   (qualify)
  Match Event Trace              = Enable   (match)
  DMA Status Trace               = Enable   (status)

>dmasswoff
DMA Sub-switch OFF Settings:
  Data Trace Point 1 Trace       = Not Trace (td1 none)
  Data Trace Point 2 Trace       = Not Trace (td2 none)
  Data Trace Point 3 Trace       = Not Trace (td3 none)
  Data Trace Point 4 Trace       = Not Trace (td4 none)
  Event Access Point 1 Trace     = Not Trace (evap1 none)
  Event Access Point 2 Trace     = Not Trace (evap2 none)
  Event Access Point 3 Trace       = Not Trace (evap3 none)
  Event Access Point 4 Trace     = Not Trace (evap4 none)
  Event Access Point 5 Trace     = Not Trace (evap5 none)
  Event Access Point 6 Trace     = Not Trace (evap6 none)
  Event Access Range 1 Trace     = Not Trace (evar1 none)
  Event Access Range 3 Trace     = Not Trace (evar3 none)
  Event Access Range 5 Trace     = Not Trace (evar5 none)
  Start Point Trace              = Not Trace (startpoint none)
  Section Event Trace            = Disable (!section)
  Qualify Event Trace            = Disable (!qualify)
  Match Event Trace              = Disable (!match)
  DMA Status Trace               = Disable (!status)
```

## sfr command

**This command cannot be used now**

[Format]

```
sfr    [reg [VAL]]
sfr2   [reg [VAL]]
```

[Parameters]

| | |
|---|---|
| reg: | Specifies an SFR register name. |
| VAL: | Specifies the value for an SFR register in hexadecimal. |

[Examples]

```
sfr P0
```

The value of the P0 register is displayed.

```
sfr PM0 0
```

The value 0h is set in the PM0 register.

## sfrfile command **This command cannot be used now**

[Format]

```
sfrfile    {/|FILENAME}
```

[Parameters]

| | |
|---|---|
| /: | Initialize the registration of SFR. |
| FILENAME: | Specifies the device file name that the NEC offers. |

[Function]

The sfrfile command extracts and registers the SFR information in a device file (FILENAME). The SFR information is used by the SFR command.
When the sfrfile command is executed, SFR information that had been registered before it extracts it is deleted.

[Example]

```
sfrfile c:¥test¥xxxxxx.800
```

Read the device file (xxxxxx.800) in the c:¥test¥ directory.

## symfile and sym commands

[Format]

```
symfile FILENAME
sym [NAME]
```

[Parameters]

| | |
|---|---|
| FILENAME: | Specifies file name. |
| NAME: | Specifies first character string in the symbols to be displayed. |

[Function]

The symfile command reads symbols from the elf file specified by the FILENAME parameter.
The symbol information which the symfile command reads is only a global symbol.
The sym command displays up to 30 symbols that have been read.

[Examples]

```
symfile c:¥test¥dry¥dry.elf
```

Symbols are read from the elf file dry.elf in the c:¥test¥dry directory.

```
sym m
```

Up to 30 symbols that begin with "m" are displayed.

## td1 .. td4 commands

[Format]

td{1|2|3|4} [LADDR [UADDR]] [asid ASID|noasid] [/del]

[Parameters]

| | |
|---|---|
| td{1\|2\|3\|4}: | Input before the condition of a command from td1 to td4 is specified. |
| LADDR: | Specifies an lower address in hexadecimal. |
| UADDR: | Specifies an upper address in hexdecimal. |
| asid ASID\|noasid: | A setup of ASID conditions |
| ASID: | Specifies ASID value in hexadecimal. |
| noasid: | ASID is not used for conditions. |
| /del: | Clears the specified address. |

[Function]

The td1 .. td4 commands set the conditions of the data access cycles to be recorded by trace.
The range of the access address is as follows.

lower address(LADDR) <= access base address <=upper address(UADDR)

If the upper address is not inputted, the same address as the lower address will be used.

[Example]

```
td1 100000 100fff
```

The access cycle of address from 100000h to 100fffh is loaded to trace

[Remark]

To display trace data in the access cycle of the area specified by td1 to td4, specify the type of access cycle in which trace data is to be output for the sswon/sswoff command.

## tenv command

[Format]

```
tenv  [tclkdiv{1|2|4|6|8|16}] [[!]ddr]
      [tdwidth{4|24}] [size{256k|512k|1m|2m|4m}] [[!]evti]
      [dtm{1|2|3|4|5|6}] [subor|suband] [[!]sss_st_off] [[!]qss_st_off]
      [[!]sss_on_dly1] [[!]sss_off_dly1]
      [lvresume{0..26/100}] [lvsuspend{2..28/102}]
```

[Parameters]

| | |
|---|---|
| tclkdiv{1\|2\|4\|6\|8\|16}: | Specifies rate of dividing frequency of the trace clock to the operation clock of CPU. Each parameter is equal to 1/1, 1/2, 1/4, 1/6, 1/8, and 1/16. |
| tdwidth{4\|24}: | Specifies the trace data width. It corresponds to 4 bits and 24 bits. |
| size{256k\|512k\|1m\|2m\|4m}: | Specifies use capacity of the trace memory. It corresponds to 256K, 512K, 1M, 2M, and 4M Bytes. |
| subor: | Specifies OR condition of the section condition and the qualify condition for a sub-switch. |
| suband: | Specify AND condition of the section condition and the qualify condition for a sub-switch. |
| [!]sss_st_off: | Starts the section sub-switch in the off state. Enter '!' to starts the sub-switch int the on state. |
| [!]qss_st_off: | Starts the qualify sub-switch in the off state. Enter '!' to starts the sub-switch int the on state. |
| [!]sss_on_dly1: | Delays setting the section sub-switch to on one instruction after the condition is satisfied. Enter ! to immediately set the sub-switch to on. |
| [!]sss_off_dly1: | Delays setting the section sub-switch to off one instruction after the condition is satisfied. Enter ! to immediately set the sub-switch to off. |
| [!]ddr: | Use the default value. |
| [!]evti: | Use the default value. |
| dtm{1\|2\|3\|4\|5\|6}: | Use the default value. |
| lvresume{0..26/100}: | Use the default value. |
| lvsuspend{2..28/102}: | Use the default value. |

[Function]

The tenv command sets the trace environment.

[Example]

```
tenv subor
```

OR of the section and qualify conditions is used as the sub-switch.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

## tsp1 and tsp2 commands

[Format]

```
tsp{1|2}    [ADDR] [asid ASID|noasid] [/del]
```

[Parameters]

| | |
|---|---|
| tsp{1\|2}: | Input before the condition of tsp1 or tsp2 is specified. |
| ADDR: | Specifies an even-numbered address in hexadecimal. |
| asid ASID\|noasid: | A setup of ASID conditions |
| ASID: | Specifies ASID value in hexadecimal. |
| noasid: | ASID is not used for conditions. |
| /del: | Clears the specified address. |

[Function]

The tsp1 and tsp2 commands specify the start points (addresses) of the two trace points.
The cycle in which the trace information is to be loaded can be changed by using the specified point. (For information on how to specify the loading condition, see the description of the sswon and sswoff commands.)

[Example]

```
tsp1 100000
```

The execution of the instruction at 100000h is specified to start point 1.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

## tmode command

[Format]
tmode

[Parameter]
None

[Function]
The tmode command displays the trace setting status.

[Example]
The default status is shown below as an example:

```
>tmode
Trace Settings (tron):
 Delay Count              = 0007ffff
 Trace Mode               = Real Time (real)
 Trace Enable Unit        = PE1/PE2/DMA (pe1 pe2 dma)
 Start Mode               = Force Start PE1/PE2/DMA (force)
 Delay Mode               = Disable (!delay)
 Ext Trigger              = Disable (noext)
Trace Env Settings :
 >--Setting for All Trace Unit (PE1/PE2/DMA)
  TRCCLK Div.             = 1/2 (tclkdiv2)
  TRCCLK Edge             = Both (ddr)
  Suspend Level           = 60 (lvsuspend60)
  Resume Level            = 20 (lvresume20)
  TDATA Width             = 24bit (tdwidth24)
  Trace Buffer            = 1M Cycles (size1m)
  Sync. by EVTI-          = Disable (!evti)
 >--Setting for Trace Unit of each Processor Element (PEx)
  Data Trace Message Type             = with PCM ,with Access-ID (dtm4)
  Sub switch                          = <section> or <qualify> (subor)
  Section Sub Switch at force start   = on (!sss_st_off)
  Qualify Sub Switch at force start   = on (!qss_st_off)
  Section Sub Switch turn on delay    = immediately (!sss_on_dly1)
  Section Sub Switch turn off delay   = immediately (!sss_off_dly1)
Trace Start Point Settings:
      Address     ASID
 tsp1 /del
 tsp2 /del
```

[Remark]
For the details of the trace, see Appendix A, "Details of Trace Functions".

## tron command

[Format]

```
Tron  [DELAY] [[!]delay] [[!]real] [[!]pe1] [[!]pe2] [[!]dma]
      [[[!]force] |[force_{1}{2}{d}]] [noext|posi|nega]
```

[Parameters]

| | |
|---|---|
| DELAY = 0..xxxx: | delay counter<br>Specifies the number of frames in memory that are to be loaded in response to a trigger, in hexadecimal. |
| [!]delay: | Specifies forced delay mode. Enter !delay to return to normal mode.<br>In forced delay mode, trace is forcibly terminated when the number of frames specified by the delay counter are traced after trace starts. In this mode, trigger events are ignored. |
| [!]real: | Specifies the execution mode during trace. real specifies the real-time execution mode. The trace information may overflow in real-time execution mode. '!' specifies the non-real-time execution mode. An overflow does not occur in this mode, but the execution speed drops. |
| [!]pe1: | Trace of PE1 is enabled. It is disabled when '!' is entered. |
| [!]pe2: | Trace of PE2 is enabled. It is disabled when '!' is entered. |
| [!]dma | Trace of DMA is enabled. It is disabled when '!' is entered. |
| [!]force: | Trace is compulsorily started by execution of the tron command. Trace of pe1, pe2, and dma is started compulsorily. The load conditions of trace information are conditions of the sswon command and the sswoff command. Even when trace is started compulsorily, the sswon command and the sswoff command are available. It is released when '!' is entered. |
| force_{1}{2}{d}: | Specifies the core that trace starts compulsorily. '1' is pe1, '2' is pe2, and 'd' is dma. |
| noext\|posi\|nega: | Specifies an external input pin (EXI0) as a trigger. |
| noext: | Does not use EXI0 as a trigger. |
| posi: | Uses the rising edge of EXI0 as a trigger. |
| nega: | Uses the falling edge of EXI0 as a trigger. |

[Function]

The tron command clears the trace buffer and the settings of trace, and begins loading trace data.

[Examples]

tron

When tron is specified using the default values, trace is forcibly started and continues until forcibly terminated. Trace data displayed after a break shows the execution status until the execution immediately before the break.

tron delay 3fffd

Trace is started in the forced delay mode (delay=on) with using the default values for other parameters. Trace data in as many cycles as specified by the delay counter value (0x3fffd) is loaded immediately after the start of execution, and trace is automatically terminated. In the forced delay mode, trigger events are ignored.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

## troff command

[Format]

troff

[Parameter]

None

[Function]

The troff command forcibly terminates the loading of trace data.

## trace command

[Format]

trace [POS] [all|cpu|pe1|pe2|dma] [asm|asmtime|asmclr] [subNN]

[Parameters]

| | |
|---|---|
| POS=±0..xxxx: | Specifies the trace display start position in hexadecimal, assuming the vicinity of a trigger cycle or the ending cycle to be 0. |
| all\| cpu\|pe1\|pe2\|dma: | Specifies the cycle in loaded trace information that is to be displayed. |
| all: | All cycles |
| cpu: | CPU(pe1,pe2) cycles only |
| dma: | DMA cycles only |
| pe1: | pe1 cycles only |
| pe2: | pe2 cycles only |
| asm\|asmtime\|asmclr: | Specifies the display type. |
| asm: | Displays assembled listing. |
| asmtime: | Displays assembled listing and Time Tag in absolute time format. |
| asmclr: | Displays assembled listing and Time Tag in relative time format. |
| subNN: | Number of instructions to be disassembled in succession from an information item to actually be loaded (hexadecimal). The initial value is 100h (sub100). |

[Function]

The trace command displays the contents of the trace buffer.
Issuing this command during trace forcibly terminates the loading process.

[Display]

```
>trace -4 asm
   Cycle  Sub      Address  Code     Instruction              EXT  Stat
  -000004 ---- PE1:1ec00014 0e010001 addi    0001h,r1,r1      1111 EXE
          0001     1ec00018 17260004 ld.h    0004h[r6],r2          SUB
          0002     1ec0001c 16020001 addi    0001h,r2,r2           SUB
          0003     1ec00020 1f060008 ld.b    0008h[r6],r3          SUB
          0004     1ec00024 1e030001 addi    0001h,r3,r3           SUB
          0005     1ec00028 1ec300ff andi    00ffh,r3,r3           SUB
          0006     1ec0002c 05ca     bnz     1ec00034h             SUB
          0007     1ec0002e 0000     nop                           SUB
  +000000 ---- PE1:         [ MTCH ]                  1111 WP
  +000002 ---- PE1:1ec00030 1e030001 addi    0001h,r3,r3      1111 EXE
          0001     1ec00034 0000     nop                           SUB
          0002     1ec00036 0f660001 st.w    r1,0000h[r6]          SUB
          0003     1ec0003a 0000     nop                           SUB
          0004     1ec0003c 17660004 st.h    r2,0004h[r6]          SUB
          0005     1ec00040 0000     nop                           SUB
          0006     1ec00042 1f460008 st.b    r3,0008h[r6]          SUB
          0007     1ec00046 0000     nop                           SUB
          0008     1ec00048 0000     nop                           SUB
          0009     1ec0004a 0000     nop                           SUB
  +000007 ---- PE1:1ec0004c e5a5     br      1ec00010h        1111 EXE (Bcond)
          0001     1e800010 0000     nop                           SUB
```

```
>trace asmtime
   Cycle  Sub      Address  Code     Instruction            EXT  Stat
  -000004 ---- PE1:1ec00014 0e010001 addi    0001h,r1,r1      1111 EXE
                             time = 000,000,005,576.0uS
          0001     1ec00018 17260004 ld.h    0004h[r6],r2          SUB
          0002     1ec0001c 16020001 addi    0001h,r2,r2           SUB
          0003     1ec00020 1f060008 ld.b    0008h[r6],r3          SUB
          0004     1ec00024 1e030001 addi    0001h,r3,r3           SUB
          0005     1ec00028 1ec300ff andi    00ffh,r3,r3           SUB
          0006     1ec0002c 05ca     bnz     1ec00034h             SUB
          0007     1ec0002e 0000     nop                           SUB
  +000000 ---- PE1:         [ MTCH ]                        1111 WP
                             time = 000,000,005,580.3uS
  +000002 ---- PE1:1ec00030 1e030001 addi    0001h,r3,r3      1111 EXE
                             time = 000,000,005,581.8uS
          0001     1ec00034 0000     nop                           SUB
          0002     1ec00036 0f660001 st.w    r1,0000h[r6]          SUB
          0003     1ec0003a 0000     nop                           SUB
          0004     1ec0003c 17660004 st.h    r2,0004h[r6]          SUB
          0005     1ec00040 0000     nop                           SUB
          0006     1ec00042 1f460008 st.b    r3,0008h[r6]          SUB
          0007     1ec00046 0000     nop                           SUB
          0008     1ec00048 0000     nop                           SUB
          0009     1ec0004a 0000     nop                           SUB
  +000007 ---- PE1:1ec0004c e5a5     br      1ec00010h        1111 EXE (Bcond)
                             time = 000,000,005,590.8uS
          0001     1e800010 0000     nop                           SUB

>trace asmclr
   Cycle  Sub      Address  Code     Instruction            EXT  Stat
  -000004 ---- PE1:1ec00014 0e010001 addi    0001h,r1,r1      1111 EXE
                             time = 000,000,000,000.0uS
          0001     1ec00018 17260004 ld.h    0004h[r6],r2          SUB
          0002     1ec0001c 16020001 addi    0001h,r2,r2           SUB
          0003     1ec00020 1f060008 ld.b    0008h[r6],r3          SUB
          0004     1ec00024 1e030001 addi    0001h,r3,r3           SUB
          0005     1ec00028 1ec300ff andi    00ffh,r3,r3           SUB
          0006     1ec0002c 05ca     bnz     1ec00034h             SUB
          0007     1ec0002e 0000     nop                           SUB
  +000000 ---- PE1:         [ MTCH ]                        1111 WP
                             time = 000,000,000,000.0uS
  +000002 ---- PE1:1ec00030 1e030001 addi    0001h,r3,r3      1111 EXE
                             time = 000,000,000,000.0uS
          0001     1ec00034 0000     nop                           SUB
          0002     1ec00036 0f660001 st.w    r1,0000h[r6]          SUB
          0003     1ec0003a 0000     nop                           SUB
          0004     1ec0003c 17660004 st.h    r2,0004h[r6]          SUB
          0005     1ec00040 0000     nop                           SUB
          0006     1ec00042 1f460008 st.b    r3,0008h[r6]          SUB
          0007     1ec00046 0000     nop                           SUB
          0008     1ec00048 0000     nop                           SUB
          0009     1ec0004a 0000     nop                           SUB
  +000007 ---- PE1:1ec0004c e5a5     br      1ec00010h        1111 EXE (Bcond)
                             time = 000,000,000,012.8uS
          0001     1e800010 0000     nop                           SUB
```

```
>trace 14 asm
  +000014 ---- DMA:--       [ DMA stat=0000000008 ]          1111 DMA STAT
  +000017 ---- DMA:1ec01000 ------00  [Byte   Read]          1111 RD DMA
  +00001c ---- DMA:1ec04000 ------00  [Byte   Write]         1111 WR DMA
  +000020 ---- PE1:1ec001a4 fd89     bnc     1ec00194h       1111 EXE (Bcond)
          0001     1ec00194 301d     mov     r29,r6               SUB
          0002     1ec00196 57060000 ld.b    0000h[r6],r10        SUB
          0003     1ec0019a 100a     mov     r10,r2               SUB
          0004     1ec0019c 7002     mov     r2,r14               SUB
          0005     1ec0019e 7288     shr     00000008h,r14        SUB
          0006     1ec001a0 05b1     bc      1ec001a6h            SUB
          0007     1ec001a2 1284     shr     00000004h,r2         SUB
  +000024 ---- DMA:1ec01000 ------01  [Byte   Read]          1111 RD DMA
  +00002a ---- DMA:1ec04000 ------01  [Byte   Write]         1111 WR DMA
  +00002e ---- DMA:1ec01002 ------02  [Byte   Read]          1111 RD DMA
  +000032 ---- PE1:1ec001a4 fd89     bnc     1ec00194h       1111 EXE (Bcond)
          0001     1ec00194 301d     mov     r29,r6               SUB
          0002     1ec00196 57060000 ld.b    0000h[r6],r10        SUB
          0003     1ec0019a 100a     mov     r10,r2               SUB
```

Cycle: Relative positions in the trace buffer are displayed in hexadecimal. The vicinity of the trigger point or the trace end frame is assumed to be 0.

Sub: Cycle numbers generated by analyzing branching and number-of-executed-instruction information.

Address: Execution addresses or bus cycle addresses are displayed.
In the case of a CPU cycle, a processor and the address are displayed in the form of the following.

PE[1|2]:address

In the case of a DMA cycle, it displays in the form of the following.

DMA:address

Code: Instruction code or bus cycle data is displayed.

Instruction: Instruction mnemonics or bus types are displayed.

EXT: The states of external input pins EXI3 to EXI0 are displayed as bit strings.

Stat: The types of trace packets(messages) on which display is based are displayed.

| | |
|---|---|
| EXE | Instruction execution |
| EXE () | Instruction execution and branching cause (() later description) |
| EXE <> | Instruction execution and condition judgment result (<> later description) |
| WR CPU | Memory write (CPU) |
| RD CPU | Memory read (CPU) |
| WR DMA | Memory write (DMA) |
| RD DMA | Memory read (DMA) |
| OTM | Task/process ID |
| WP | Watch point<br>Display the watch cause (later description) in the Code display part. |
| ERR | Error code<br>Display the error code in the Code display part. |
| DMA STAT | DMA status |

"()" indicates the following character strings. It indicates an instruction or an event that has caused branch.

| | |
|---|---|
| NMI/INT | By occurrence of interrupt |
| EXP/TRAP | By occurrence of exception |
| RETI | By corresponding instruction |

| | |
|---|---|
| JMP | By corresponding instruction |
| JR | By corresponding instruction |
| JARL | By corresponding instruction |
| Bcond | By corresponding instruction |
| CALLT | By corresponding instruction |
| SWITCH | By corresponding instruction |
| DISPOSE | By corresponding instruction |
| CTRET | By corresponding instruction |
| SYSCALL | By corresponding instruction |

"<>" indicates the following character strings. It indicates that there was a condition judgment.

| | |
|---|---|
| T | True |
| NT | Not true |

The watch cause indicates the following character strings.

| | |
|---|---|
| WP1..4 | Watch point 1..4 |
| MTCH | Match event |
| Q-ON | Qualify sub-switch ON |
| Q-OFF | Qualify sub-switch OFF |
| S-ON | Section sub-switch ON |
| S-OFF | Section sub-switch OFF |

time = Displays Time Tag

**Remark** The Time Tag reflects a value when the CPU outputs branch information. The output of branch information has some delay from the time of actual execution, and the delay is not constant. Thus, the measurement value of the Time Tag has some error. Especially, please ignore the measurement result immediately after the execution, as it is unreliable.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

## ftrace command

[Format]

```
ftrace   statpos endpos filname [trace_options]
```

[Parameters]

| | |
|---|---|
| statpos: | Start trace position to be written into a file |
| endpos: | End trace position to be written into a file |
| filname: | Input a file name |
| trace_options: | The following parameters are available. The meaning is the same as for the trace command.<br>[all\|cpu\|pe1\|pe2\|dma] [asm\|asmtime\|asmclr] [subNN] |

[Function]

The ftrace command writes the contents of the trace buffer into a file.

[Note]

Carefully enter the parameters for this command because the command cannot be canceled once executed.　If a wide range is specified, processing takes time.

## tdata_dly command

[Format]

```
tdata_dly    [off|small|medium|large]
```

[Parameters]

| | |
|---|---|
| off: | Not adjust |
| small: | The minimum adjustment |
| medium: | The medium adjustment (default) |
| large: | The maximum adjustment |

[Function]

The tdata_dly command adjusts the setup time between a trace clock and trace data. The shortest setup time is “off”. The longest setup time is “large”.

An actual setup value must depend on the RTE-xxxx-TP unit and the cable to be used. Please check the specification of the unit.

[Remark]

You need not change from an initial value usually.

However, the adjustment might become necessary according to the state of CPU and the board.

## dmatd1...dmatd4 command

[Format]

```
dmatd{1|2|3|4}    [LADDR [UADDR]] [ch CHNO|allch]
                  [[!]csext] [[!]cspe1] [[!]cspe2] [/del]
```

[Parameters]

| | |
|---|---|
| dmatd{1\|2\|3\|4}: | Input before the condition of dmatd1 to dmatd4 is specified. |
| LADDR: | Specifies an lower address in hexadecimal. |
| UADDR: | Specifies an upper address in hexdecimal. |
| ch CHNO\|allch: | Specifies an DMA channel condition |
| CHNO: | Specifies the DMA channel in hexadecimal.<br>0...7F: DTS channnel 0...127<br>80...FE: DMAC channel 0...126 |
| allch: | Specifies that doesn't use the DMA channel condition. |
| [!]csext: | Specifies the chip selection condition (external resource). It is not set when '!' is entered. |
| [!]cspe1: | Specifies the chip selection condition (PE1 access). It is not set when '!' is entered. |
| [!]cspe2: | Specifies the chip selection condition (PE2 access). It is not set when '!' is entered. |
| /del: | Clears the specified condition. |

[Function]

The td1 .. td4 commands set the conditions of the DMA access cycles to be recorded by trace.
The range of the access address is as follows.
    lower address(LADDR) <= access base address <=upper address(UADDR)
If the upper address is not inputted, the same address as the lower address will be used.

[Example]

```
dmatd1 100000 100fff
```

The DMA access cycle of address from 100000h to 100fffh is loaded to trace

[Remark]

To display trace data in the access cycle of the area specified by dmatd1 to dmatd4, specify the type of access cycle in which trace data is to be output for the dmasswon/dmasswoff command.

## dmatenv command

[Format]

```
dmatenv   [subor|suband] [[!]sss_st_off] [[!]qss_st_off]
          [[!]sss_on_dly1] [[!]sss_off_dly1] [dtm{1|2|3|4|5}]
```

[Parameters]

| | |
|---|---|
| subor: | Specifies OR condition of the section condition and the qualify condition for a sub-switch. |
| suband: | Specify AND condition of the section condition and the qualify condition for a sub-switch. |
| [!]sss_st_off: | Starts the section sub-switch in the off state. Enter '!' to starts the sub-switch int the on state. |
| [!]qss_st_off: | Starts the qualify sub-switch in the off state. Enter '!' to starts the sub-switch int the on state. |
| [!]sss_on_dly1: | Delays setting the section sub-switch to on one instruction after the condition is satisfied. Enter ! to immediately set the sub-switch to on. |
| [!]sss_off_dly1: | Delays setting the section sub-switch to off one instruction after the condition is satisfied. Enter ! to immediately set the sub-switch to off. |
| dtm{1|2|3|4|5}: | Use the default value. |

[Function]

Set the environment of the DMA trace.

[Example]

```
dmatenv subor
```

OR of the section and qualify conditions is used as the sub-switch.

## dmatsp1, dmatsp2 commands

[Format]

```
dmatsp{1|2}    [ADDR] [ch CHNO|allch] [[!]csext] [[!]cspe1] [[!]cspe2] [/del]
```

[Parameters]

| | |
|---|---|
| dmatsp{1\|2}: | Input before the condition of dmatsp1 or dmatsp2 is specified. |
| ADDR: | Specifies addresses in hexadecimal number. |
| ch CHNO\|allch: | Specifies an DMA channel condition |
| CHNO: | Specifies the DMA channel in hexadecimal.<br>0...7F: DTS channnel 0...127<br>80...FE: DMAC channel 0...126 |
| allch: | Specifies that doesn't use the DMA channel condition. |
| [!]csext: | Specifies the chip selection condition (external resource). It is not set when '!' is entered. |
| [!]cspe1: | Specifies the chip selection condition (PE1 access). It is not set when '!' is entered. |
| [!]cspe2: | Specifies the chip selection condition (PE2 access). It is not set when '!' is entered. |
| /del: | Specifies deletion of a condition. |

[Function]

The dmatsp1 and dmatsp2 commands specify the start points (addresses) of the two trace points.
The cycle in which the trace information is to be loaded can be changed by using the specified point. (For information on how to specify the loading condition, see the description of the dmasswon and dmasswoff commands.)

[Example]

```
dmatsp1 100000
```

The execution of the instruction at 100000h is specified to start point 1.

## wp command

[Format]

```
wp{1|2|3|4}    [ADDR [AMASK]] [data DATA [DMASK]] [asid ASID|noasid] [aeq|aneq]
               [deq|dneq][exec|read|write|accs] [byte|hword|word|nosize]
wp{1|2|3|4}    /del
```

[Parameters]

| | |
|---|---|
| wp{1\|2\|3\|4}: | Input before the condition of wp1 to wp4 is specified. |
| ADDR [AMASK]: | Specifies an address condition. |
| ADDR: | Specifies addresses in hexadecimal number. |
| AMASK: | Specifies the mask data of an address in hexadecimal. Bits that are 1 will not be compared. |
| data DATA [DMASK]: | Specifies a data condition. |
| DATA: | Specifies data in hexadecimal number. |
| DMASK: | Specifies the mask data of data in hexadecimal. Bits that are 1 will not be compared. |
| asid ASID\|noasid: | For future expansion. Use “noasid”. |
| aeq\|aneq: | Specifies an address comparison condition. |
| aeq: | Compares address for equality. |
| aneq: | Compares address for non-equality. |
| deq\|dneq: | Specifies a data comparison condition. |
| deq: | Compares data for equality. |
| dneq: | Compares data for non-equality. |
| exec\|read\|write\|accs: | Specifies a cycle condition. |
| exec: | Specifies an executable address. A data condition is ignored. |
| read: | Specifies a read cycle. |
| write: | Specifies a write cycle. |
| accs: | Specifies a read or write cycle. |
| byte\|hword\|word\|nosize: | Specifies access size. |
| byte: | Specifies byte access (8 bits). |
| hword: | Specifies half-word access (16 bits). |
| word: | Specifies word access (32 bits). |
| nosize: | Specifies invalidity. |
| wp{1\|2\|3\|4} /del: | Clears a condition. |
| /del: | Specifies deletion of a condition. |

[Function]

These commands set or delete watch points.
Use it on the event condition and the trace condition.

[Examples]

wp1 1000 aeq exec

A watch point for execution of address 1000h is set.

wp2 1000 data 5555 0 aeq deq read hword

Watch point occurs when 5555h is read in hword from address 1000h.

wp1 /del

The condition set by wp1 is deleted.

[Remark]

The resources of the wp command are the same as the resources of the abp command. The resources currently used by the abp command cannot be used by the wp command.
The cycle conditions which can be used for trace conditions are only "exec".

## time command

[Format]

time

[Parameter]

None

[Function]

The time command displays the time as the result of execution time measurement. The execution time measurement timer is initialized each time the CPU starts execution and is keeping the time during the execution of the CPU. The frequency of the JTAGCLK divided by 2 is used as the measurement clock frequency. The time converted to ns units is displayed.
Effective counter is 31-bit. When JTAGCLK is 25MHz, maximum count is about 160 seconds.

[Remark]

The measurement time includes the overhead times (several clocks) at the start of execution and breaks.

[Example]

```
>time
 Time = 10,320 (ns) (12.500000MHz) [Counter=00000081]        <---- when JTAGCLK=25MHz
                     |                   |_Counter value (hexadecimal)
                     |_Measurement clock frequency (JTAGCLK*1/2)
```

## fread, fwrite, ffill, fload, and fsave commands

[Format]

| | |
|---|---|
| fread | [ADDR [LENGTH]] |
| fwrite | ADDR DATA0[ DATA1[ DATA2 [ DATA3…]]] |
| ffill | ADDR LENGTH DATA |
| fload | ADDR FILENAME |
| fsave | ADDR LENGTH FILENAME |

[Parameter]

| | |
|---|---|
| ADDR: | Specifies an address in hexadecimal. |
| LENGTH: | Specifies the number of bytes to be read. |
| DATA: | Specifies the data to be write in hexadecimal. |
| FILENAME: | Specifies the file name |

[Function]

fread [ADDR [LENGTH]]

The memory range specified by ADDR and LENGTH are read and shown. The method of memory read is a fly-by.

fwrite ADDR DATA0[ DATA1[ DATA2 [ DATA3…]]]

Each data is written to the memory sequentially (data0,data1,data2…) specified by ADDR. Writing data size is set by ACC command. The method of memory write is a fly-by.

ffill ADDR LENGTH DATA

The memory is filled up with DATA by assigned ADDR and LENGTH. Writing data size is set by ACC command. The method of memory write is a fly-by.

fload ADDR FILENAME

The contents of FILENAME are downloaded to the ADDR. The method of memory write is a fly-by.

fsave ADDR LENGTH FILENAME

The memory data specified by ADDR and LENGTH are save to the FILENAME's file. The method of memory read is a fly-by.

[Remark]

These memory access commands can be used while a program executes. The accessible range is only a area which CPU approved.

When the software break point address is read during program execution, not a program code but a break code instruction is displayed.

## ver command

[Format]

```
ver
```

[Parameter]

None

[Function]

The ver command displays the version of KIT-V850E2/MN4-TP.